```
import os
import keras
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import itertools

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
import matplotlib

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
train = pd.read_csv(r'C:\Users\Nithishma\Desktop\train.csv')
print('Shape of the train data with all features:', train.shape)
train = train.select_dtypes(exclude=['object'])
print("")
print('Shape of the train data with numerical features:', train.shape)
train.drop('Id',axis = 1, inplace = True)
train.fillna(0,inplace=True)
test = pd.read_csv(r'C:\Users\Nithishma\Desktop\test.csv')
test = test.select_dtypes(exclude=['object'])
ID = test.Id
test.fillna(0,inplace=True)
test.drop('Id',axis = 1, inplace = True)


print("")

print("List of features contained our dataset:",list(train.columns))

from sklearn.ensemble import IsolationForest


clf = IsolationForest(max_samples = 100, random_state = 42)

clf.fit(train)

y_noano = clf.predict(train)

y_noano = pd.DataFrame(y_noano, columns = ['Top'])

y_noano[y_noano['Top'] == 1].index.values


train = train.iloc[y_noano[y_noano['Top'] == 1].index.values]

train.reset_index(drop = True, inplace = True)

print("Number of Outliers:", y_noano[y_noano['Top'] == -1].shape[0])

print("Number of rows without outliers:", train.shape[0])

import warnings

warnings.filterwarnings('ignore')


col_train = list(train.columns)

col_train_bis = list(train.columns)


col_train_bis.remove('SalePrice')


mat_train = np.matrix(train)

mat_test  = np.matrix(test)

mat_new = np.matrix(train.drop('SalePrice',axis = 1))
```

```
mat_y = np.array(train.SalePrice).reshape((1314,1))


prepro_y = MinMaxScaler()
prepro_y.fit(mat_y)


prepro = MinMaxScaler()
prepro.fit(mat_train)


prepro_test = MinMaxScaler()
prepro_test.fit(mat_new)


train = pd.DataFrame(prepro.transform(mat_train),columns = col_train)
test  = pd.DataFrame(prepro_test.transform(mat_test),columns = col_train_bis)
train.head()
COLUMNS = col_train
FEATURES = col_train_bis
LABEL = "SalePrice"


# Columns
feature_cols = FEATURES


# Training set and Prediction set with the features to predict
training_set = train[COLUMNS]
prediction_set = train.SalePrice


# Train and Test
x_train, x_test, y_train, y_test = train_test_split(training_set[FEATURES] , prediction_set, test_size=0.33, random_state=42)
```

```
y_train = pd.DataFrame(y_train, columns = [LABEL])
training_set = pd.DataFrame(x_train, columns = FEATURES).merge(y_train, left_index = True,
right_index = True)
training_set.head()

# Training for submission
training_sub = training_set[col_train]
y_test = pd.DataFrame(y_test, columns = [LABEL])
testing_set = pd.DataFrame(x_test, columns = FEATURES).merge(y_test, left_index = True,
right_index = True)
testing_set.head()
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor

seed = 7
np.random.seed(seed)

# Model
model = Sequential()
model.add(Dense(200, input_dim=36, kernel_initializer='normal', activation='relu'))
model.add(Dense(100, kernel_initializer='normal', activation='relu'))
model.add(Dense(50, kernel_initializer='normal', activation='relu'))
model.add(Dense(25, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
# Compile model
model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adadelta())
```

```
feature_cols = training_set[FEATURES]
labels = training_set[LABEL].values

model.fit(np.array(feature_cols), np.array(labels), epochs=100, batch_size=10)
# Evaluation on the test set created by train_test_split
model.evaluate(np.array(feature_cols), np.array(labels))
feature_cols_test = testing_set[FEATURES]
labels_test = testing_set[LABEL].values

y = model.predict(np.array(feature_cols_test))
predictions = list(itertools.islice(y, testing_set.shape[0]))
predictions = prepro_y.inverse_transform(np.array(predictions).reshape(434,1))
reality = pd.DataFrame(prepro.inverse_transform(testing_set), columns = [COLUMNS]).SalePrice
y_predict = model.predict(np.array(test))

def to_submit(pred_y,name_out):
    y_predict = list(itertools.islice(pred_y, test.shape[0]))
y_predict=pd.DataFrame(prepro_y.inverse_transform(np.array(y_predict).reshape(len(y_predict),1)), columns = ['SalePrice'])
    y_predict = y_predict.join(ID)
    y_predict.to_csv(name_out + '.csv',index=False)

to_submit(y_predict, "Realestate_prices")
```